AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1-31. (cancelled)

32. (currently amended) A method of a development and build environment for packaged software delivery in a distributed network of nodes, the method comprising the computer-implemented steps of the development and build environment:

compiling source code files into one or more executable file modules;

wherein each of the one or more modules contains an image for a process or a dynamically linked library (DLL);

creating a software package that comprises the one or more modules, wherein the software package is delivered to the nodes in the distributed network;

wherein the software package is created based on at least one of a feature, characteristic, or purpose;

creating metadata for a first module, of the one or more modules, that includes any module information such as the first module's: binary signature, name, directory path, and characteristics;

inserting the metadata of the first module into the software package; and

gathering application program interface (API) dependency information for the first module, wherein the first module can provide and use at least one API, by

(a) receiving a list of dependent modules for a given process or DLL ~~module~~ of the first module;

(b) storing, in the metadata of the first module, dependency information for the dependent modules in the list, wherein the dependency information includes API names and versions that the process or DLL ~~module~~ depends on;

(c) collecting additional dependency information documented in one or more ~~additional~~ modules specifications that are separate from the list of dependent modules, wherein the additional dependency information includes API names and versions that the process or DLL ~~module~~ depends on; and

(d) storing the additional dependency information in the metadata of the first module.

33. (currently amended) A method as recited in Claim 32, wherein a linker creates the list of dependent modules for the given process or DLL ~~module~~ and places the list in the metadata of the first module.

34. (previously presented) A method as recited in Claim 32, further comprising the steps of:
creating metadata for each API; and
inserting the API metadata into the software package, wherein metadata for an API includes, but is not limited to: the API's name and version.

35. (previously presented) A method as recited in Claim 32, further comprising the step of:
calculating a binary signature for each module of the one or more modules and inserting the binary signature into the respective module's metadata;
wherein each unique version of a module will have a unique binary signature.

36. (previously presented) A method as recited in Claim 32, further comprising the steps of:
creating metadata for the software package that includes any package information such as the software package's: name, build date, and characteristics; and

inserting the metadata of the software package into the software package.

37. (currently amended) A method of a development and build environment for packaged software delivery in a distributed network of nodes, the method comprising the computer-implemented steps of the development and build environment:

compiling source code files into one or more executable file modules;

wherein each of the one or more modules contains an image for a process or a dynamically linked library (DLL);

creating a software package that comprises the one or more modules, wherein the software package is delivered to the nodes in the distributed network;

creating metadata for a first module, of the one or more modules, that includes any module information such as the first module's: binary signature, name, directory path, and characteristics;

inserting the metadata of the first module into the software package; and

gathering application program interface (API) dependency information for the first module, wherein the first module can provide and use at least one API, by

(a) receiving a list of dependent modules for a given process or DLL ~~module~~ of the first module; and

(b) storing, in the metadata of the first module, dependency information for the dependent modules in the list, wherein the dependency information includes API names and versions that the process or DLL ~~module~~ depends on.

38. (currently amended) A method as recited in Claim 37, wherein a linker creates the list of dependent modules for the given process or DLL ~~module~~ and places the list in the metadata of the first module.

39. (currently amended) A method as recited in Claim 37, further comprising the step of:

collecting additional dependencies documented in one or more ~~additional~~ module specifications that are separate from the list of dependent modules and placing [[them]] the additional dependencies into the metadata of the first module;

wherein the additional dependencies documented in each module lists API names and versions that the process or DLL ~~module~~ depends on.

40. (previously presented) A method as recited in Claim 37, further comprising the steps of:

creating metadata for each API; and

inserting the API metadata into the software package, wherein metadata for an API includes, but is not limited to: the API's name and version.

41. (previously presented) A method as recited in Claim 37, further comprising the step of:

calculating a binary signature for each module of the one or more modules and inserting the binary signature into the respective module's metadata;

wherein each unique version of a module will have a unique binary signature.

42. (previously presented) A method as recited in Claim 37, wherein packages are created based on at least one of a feature, characteristic, or purpose.

43. (previously presented) A method as recited in Claim 37, further comprising the steps of:

creating metadata for the software package that includes any package information such as the package's: name, build date, and characteristics; and

inserting the metadata of the software package into the software package.

44. (currently amended) An apparatus for a development and build environment for packaged software delivery in a distributed network of nodes, comprising:

means for compiling source code files into one or more executable file modules;

wherein each of the one or more modules contains an image for a process or a dynamically linked library (DLL);

means for creating a software package that comprises the one or more modules, wherein the software package is delivered to the nodes in the distributed network;

means for creating metadata for a first module, of the one or more modules, that includes any module information such as the first module's: binary signature, name, directory path, and characteristics;

means for inserting the metadata of the first module into the software package; and

means for gathering application program interface (API) dependency information for the first module, wherein the first module can provide and use at least one API, by

(a) receiving a list of dependent modules for a given process or DLL ~~module~~ of the first module; and

(b) storing, in the metadata of the first module, dependency information for the dependent modules in the list, wherein the dependency information includes API names and versions that the process or DLL ~~module~~ depends on.

45. (currently amended) An apparatus as recited in Claim 44, further comprising:

a linker, wherein said linker creates the list of dependent modules for the given process or DLL ~~module~~ and places the list in the metadata of the first module.

46. (currently amended) An apparatus as recited in Claim 44, further comprising:

means for collecting additional dependencies documented in one or more ~~additional~~ module specifications that are separate from the list of dependent modules and placing [[them]] the additional dependencies into the metadata of the first module;

wherein the additional dependencies documented in each module lists API names and versions that the process or DLL ~~module~~ depends on.

47. (previously presented) An apparatus as recited in Claim 44, further comprising:

means for creating metadata for each API; and

means for inserting the API metadata into the software package, wherein metadata for an API includes, but is not limited to: the API's name and version.

48. (previously presented) An apparatus as recited in Claim 44, further comprising:

means for calculating a binary signature for each module of the one or more modules and inserting the binary signature into the respective module's metadata, wherein each unique version of a module will have a unique binary signature.

49. (previously presented) An apparatus as recited in Claim 44, wherein packages are created based on at least one of a feature, characteristic, or purpose.

50. (previously presented) An apparatus as recited in Claim 44, further comprising:

means for creating metadata for the software package that includes any package information such as the software package's: name, build date, and characteristics; and

means for inserting the metadata of the software package into the software package.

51. (currently amended) A computer-readable medium, tangibly embodied on a physical storage medium, wherein the computer-readable medium is one of a volatile medium or a non-volatile medium, the computer-readable medium carrying one or more sequences of instructions for a development and build environment for packaged software delivery in a distributed network of nodes, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of the development and build environment:

compiling source code files into one or more executable file modules;

wherein each of the one or more modules contains an image for a process or a dynamically linked library (DLL);

creating a software package that comprises the one or more modules, wherein the software package is delivered to the nodes in the distributed network;

creating metadata for a first module, of the one or more modules, that includes any module information such as the module's: binary signature, name, directory path, and characteristics;

inserting the metadata of the first module into the software package; and

gathering application program interface (API) dependency information for the first module, wherein the first module can provide and use at least one API, by

(a) receiving a list of dependent modules for a given process or DLL ~~module~~ of the first module; and

(b) storing, in the metadata of the first module, dependency information for the dependent modules in the list, wherein the dependency information includes API names and versions that the process or DLL ~~module~~ depends on.

52. (currently amended) A computer-readable medium as recited in Claim 51, wherein a linker creates the list of dependent modules for the given process or DLL ~~module~~ and places the list in the metadata of the first module.

53. (currently amended) A computer-readable medium as recited in Claim 51, wherein the one or more stored sequences of instructions which, when executed by the one or more processors, further cause the one or more processors to carry out the step of:

collecting additional dependencies documented in one or more ~~additional~~ module specifications that are separate from the list of dependent modules and placing [[them]] the additional dependencies into the metadata of the first module;

wherein the additional dependencies documented in each module lists API names and versions that the process or DLL ~~module~~ depends on.

54. (previously presented) A computer-readable medium as recited in Claim 51, wherein the one or more stored sequences of instructions which, when executed by the one or more processors, further cause the one or more processors to carry out the steps of:

creating metadata for each API; and

inserting the API metadata into the software package, wherein metadata for an API includes, but is not limited to: the API's name and version.

55. (previously presented) A computer-readable medium as recited in Claim 51, wherein the one or more stored sequences of instructions which, when executed by the one or more processors, further cause the one or more processors to carry out the step of:
calculating a binary signature for each module of the one or more modules and inserting the binary signature into the respective module's metadata, wherein each unique version of a module will have a unique binary signature.

56. (previously presented) A computer-readable medium as recited in Claim 51, wherein packages are created based on at least one of a feature, characteristic, or purpose.

57. (previously presented) A computer-readable medium as recited in Claim 51, wherein the one or more stored sequences of instructions which, when executed by the one or more processors, further cause the one or more processors to carry out the steps of:
creating metadata for the software package that includes any package information such as the software package's: name, build date, and characteristics; and
inserting the metadata of the software package into the software package.

58. (currently amended) An apparatus running a development and build environment for packaged software delivery in a distributed network of nodes, the apparatus comprising:
a network interface that is coupled to a data network for receiving one or more packet flows therefrom;

a processor;

one or more stored sequences of instructions which, when executed by the processor,

cause the processor to carry out the steps of:

compiling source code files into one or more executable file modules;

wherein each of the one or more modules contains an image for a process or a dynamically linked library (DLL);

creating a software package that comprises the one or more modules, wherein the software package is delivered to the nodes in the distributed network;

wherein packages are created based on at least one of a feature, characteristic, or purpose;

creating metadata for a first module, of the one or more modules, that includes any module information such as the first module's: binary signature, name, directory path, and characteristics;

inserting the metadata of the first module into the software package; and

gathering application program interface (API) dependency information for the first module, wherein the first module can provide and use at least one API, by

(a) receiving a list of dependent modules for a given process or DLL ~~module~~ of the first module;

(b) storing, in the metadata of the first module, dependency information for the dependent modules in the list, wherein the dependency information includes API names and versions that the process or DLL ~~module~~ depends on;

(c) collecting additional dependency information documented in one or more ~~additional~~ modules specifications that are separate from the list of dependent modules, wherein dependency information includes API names and versions that the process or DLL ~~module~~ depends on; and

(d) storing the additional dependency information in the metadata of the first module.

59. (currently amended) An apparatus as recited in Claim 58, wherein a linker creates the list of dependent modules for the given process or DLL ~~module~~ and places the list in the metadata of the first module.

60. (previously presented) An apparatus as recited in Claim 58, wherein the one or more stored sequences of instructions which, when executed by the processor, further cause the processor to carry out the steps of:

creating metadata for each API; and

inserting the API metadata into the software package, wherein metadata for an API includes, but is not limited to: the API's name and version.

61. (previously presented) An apparatus as recited in Claim 58, wherein the one or more stored sequences of instructions which, when executed by the processor, further cause the processor to carry out the step of:

calculating a binary signature for each module of the one or more modules and inserting the binary signature into the respective module's metadata, wherein each unique version of a module will have a unique binary signature.

62. (previously presented) An apparatus as recited in Claim 58, wherein the one or more stored sequences of instructions which, when executed by the processor, further cause the processor to carry out the steps of:

creating metadata for the software package that includes any package information such as the software package's: name, build date, and characteristics; and

inserting the metadata of the software package into the software package.